Programmation avancée

Examen intermédiaire

jeudi 20 novembre 2008

Nom : _____

Prénom : _____

Section : _____

Vos points sont *précieux*, ne les gaspillez pas !

Votre nom Le travail qui ne peut pas vous être attribué est perdu: écrivez votre nom sur chaque feuille que vous rendez.

Votre temps Tous les points ne sont pas égaux. En effet, nous ne pensons pas que tous les exercices ont la même difficulté, même s'ils ont le même nombre de points.

Votre attention La donnée de chaque exercice est précisément formulée, et parfois subtile. Si vous ne la comprenez pas, vous ne pourrez pas en tirer tous les points.

Exercice	**Points**	**Points obtenus**
1	10	
2	10	
3	10	
Total	30	

Exercice 1 : Chaines de caractères (10 points)

Le trait `Text` déclare un type de données pour des chaines de caractères. La définition de `Text` est la suivante, et ne peut pas être modifiée.

```
trait Text {
  def charAt(index: Int): Char
  def append(that: Text): Text
  def length: Int
}
```

Dans `Text`, l'implantation de `append` doit être de complexité $O(1)$, c'est-à-dire qu'elle ne contient ni boucle, ni récursion, ni aucun appel à des méthodes qui ne sont pas elle-même en $O(1)$. La complexité des autres opérations n'est pas définie.

Votre tâche pour cet exercice

1. Vous implantez complètement le type de données déclaré par le trait `Text`, uniquement à l'aide d'une ou plusieurs sous-classes concrètes.
2. Vous implantez la fonction `mkText` qui, pour un `String` donné retourne le `Text` correspondant.

Exercice 2 : Boucle `for` (10 points)

Écrivez en Scala une fonction `cfor` qui simule la boucle *for* de C. On doit pouvoir utiliser `cfor` de la façon suivante.

```
var i = 0
cfor (i = 42, i > 0, i -= 1) {
 println("Step: " + i)
}
```

Le comportement du programme Scala ci-dessus doit être identique à celui du programme C suivant.

```
int i = 0;
for (i = 42; i > 0; i--) {
 printf("Step: %d", i);
}
```

Vous remarquerez que la boucle *for* de C requiert la déclaration de la variable d'itération avant la définition de la boucle.

Votre tâche pour cet exercice

1. Vous implantez la fonction `cfor` en Scala.
2. Vous proposez une traduction du programme C suivant en Scala, en utilisant votre implantation de `cfor`.

   ```
   int i, j = 0;
   for (i = 42, j = 1; i > 0; i--, j++) {
    printf("Step: %d", i+j);
   }
   ```

Exercice 3 : Aidez le CERN (10 points)

Un des buts évidents du LHC est de créer des trous noirs. Le *BlackTrack* est un instrument qui mesure les occurrences de trous noirs pendant une expérience. Il génère une `List[Int]` qui décrit les instants auxquels un trou noir est détecté. Chaque élément n de la liste correspond à la détection d'un trou noir n millisecondes après le début de l'expérience. Par ailleurs, il ne peut y avoir deux détections inscrites dans la liste pour le même n et les éléments de la liste sont triés du plus petit au plus grand.

De temps en temps, le *BlackTrack* mesure deux fois le même trou noir, qui apparait donc à double dans la liste de détections. Heureusement, il est facile d'identifier ces erreurs : chaque détection qui à lieu moins de 4 millisecondes après la précédente est considérée comme erronée.

Notez que des détections erronées peuvent, elle-mêmes, amener d'autres détections à être considérées comme erronées. C'est-à-dire que la mesure `List(5, 7, 10)` contient deux détections erronées à 7 et à 10 millisecondes.

Votre tâche pour cet exercice

1. Vous implantez la fonction `bad` qui prend en argument une mesure du *BlackTrack* sous forme de `List[Int]` et qui retourne une `List[Int]` contenant uniquement les détections erronées de l'argument.